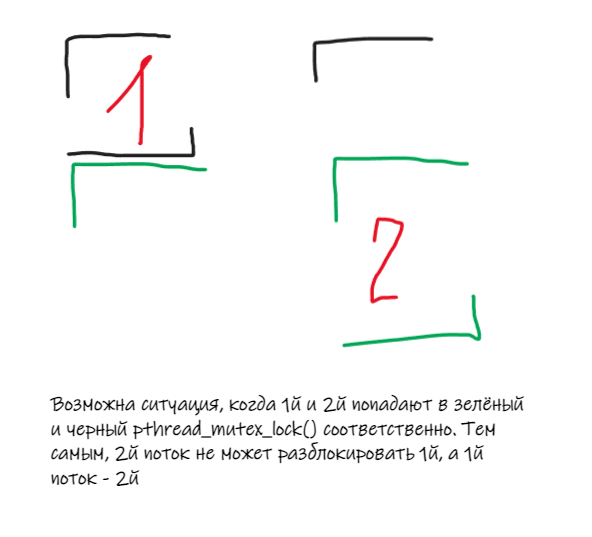
* Иллюстрация проблемы 2х мьютексов:



Все, что старался показать на занятии – поиск варианта с тремя мьютексами. Слишком сложно было проиллюстрировать, как я пришёл к правильной комбинации – это было больше похоже на логику плюс эксперименты.

Поэтому лучше будет, если покажу, что существующий вариант работает корректно.

1. *В Р (parent) lock m1 до создания С, в С (child) lock m1 и lock m2.*  Это вызовет блокировку С на m1.Таким образом мы добились того, что первым будет Р: *вывод Р, lock m3, unlock m1, lock 2 ->* блокировка P, т.к. на С lock 2*.*
2. *В Р unlock m1 –>* разблокировка С: *вывод С, unlock m2, lock m3 ->* блокировка C;
3. *В С unlock m2 ->* разблокировка Р: *unlock m3(Разблокировка С), lock m1, unlock m2 …*
4. *В P unlock m3 ->* разблокировка С:  *unlock m1, lock m2…;*

В *3), 4)* идёт ветвление. Рассмотрим варианты следующие из *unlock m3*:

А) Р lock m1 обгоняет C unlock m1, значит unlock m2 будет раньше lock m2 -> получили lock m2

Б) C unlock m1 быстрее P lock m1, значит lock m2 раньше unlock m2 -> получили lock m1

Если А), то Р сделает вывод и заблокируется в m2 и потом пункт *2)*. Причем Р выведет раньше, т.к. в А) Р обгоняет С == «ТОРМОЖЕНИЕ БЫСТРОГО Р»

Если Б), то заблокируется С, получим пункт *1)* == «ТОРМОЖЕНИЕ БЫСТРОГО С»

Тем самым показываю, что получаем корректную работу цикла.

**Конец доказательства**

Как ещё можно убедиться в корректности программы? Расставлять разные sleep()/usleep() между мьютексами. Задержки никак не повлияют на порядок вывода, лишь увеличат время исполнения программы.